

\$3.00

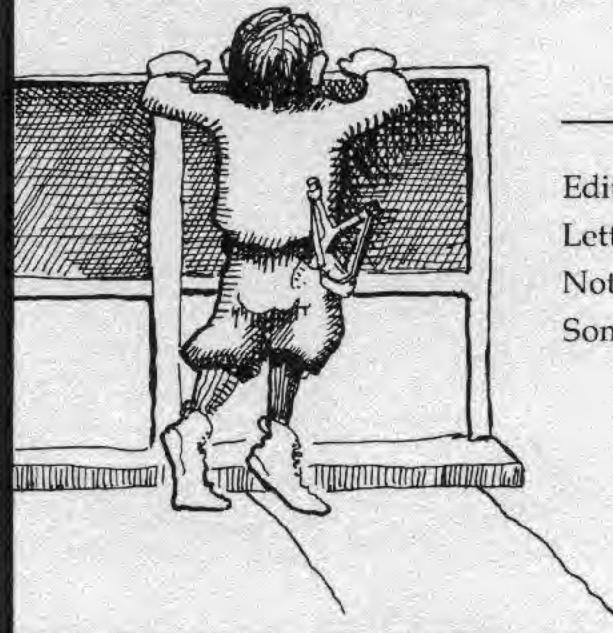


September 1981

No. 2

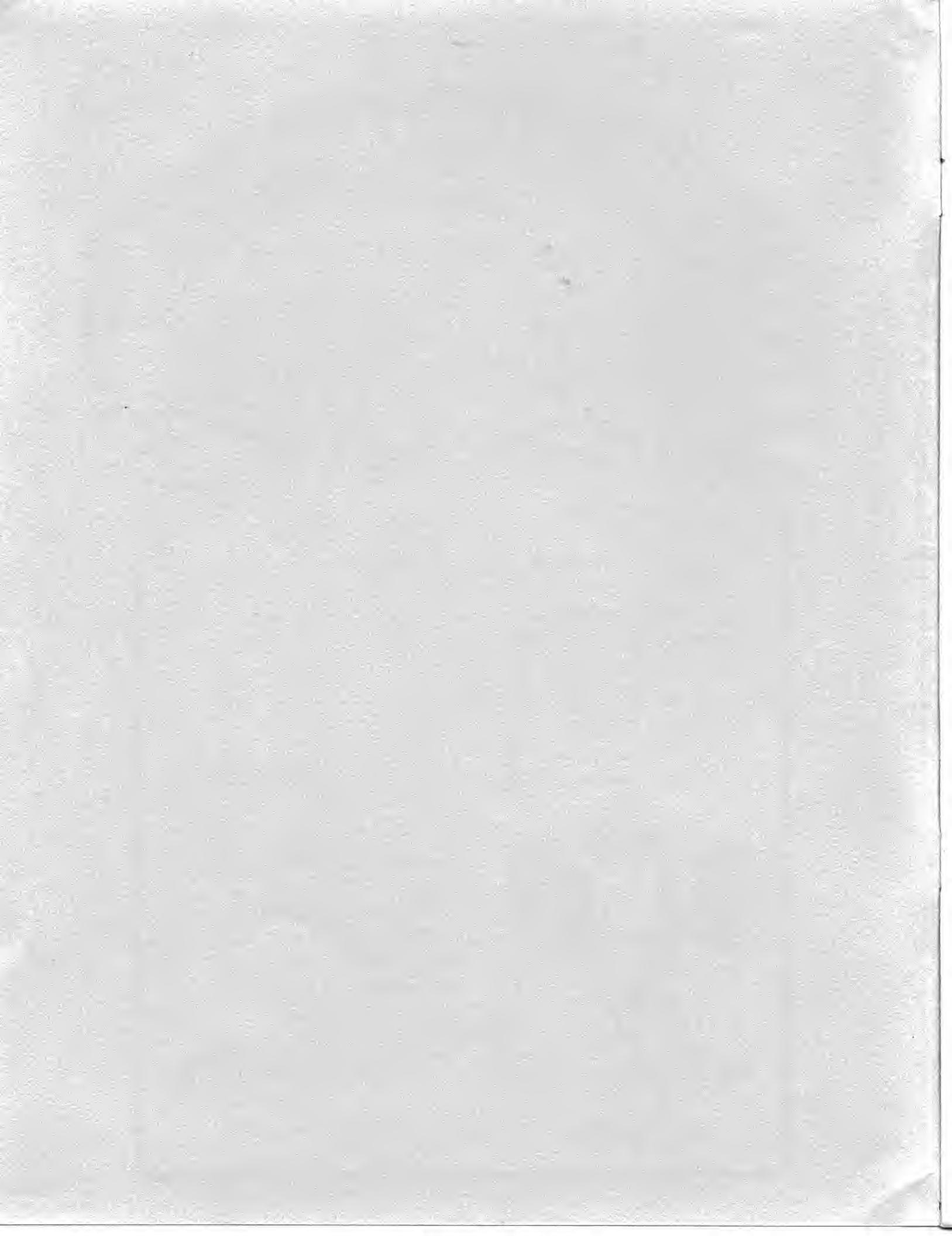
TABLE OF CONTENTS

Supporting a Language.....	2
Parallel Print Driver & Listing	3
Disk Drive Motor Control	5
Jumpering the Wild Shugart.....	6
More Power Supplies.....	7
Direct Input Routine & Listing.....	7
Program Storage Above PFM & Listing	8



REGULAR FEATURES

Editorial	1
Letters.....	2
Notes from Garland	4
Something New	7



MICRO CORNUCOPIA
11740 N.W. West Road
Portland, Oregon 97229
503-645-3253

Editor & Publisher
David J. Thompson
Technical Editor
Ruth Fredine-Burt
Graphic Design
Sandra Thompson
Typography
Patti Morris & Martin White
Irish Setter
Cover Illustration
Gerald Torrey

MICRO CORNUCOPIA is published six times a year by Micro Cornucopia of Oregon, 11740 N.W. West Road, Portland, Oregon 97229.

SUBSCRIPTION RATES:

1 yr. (6 issues)	\$12.00
1 yr. (Canada)	\$15.00
1 yr. (other foreign)	\$20.00

All subscription orders payable in United States funds only, please.

ADVERTISING RATES: Available on request.

CHANGE OF ADDRESS: Please send old label and new address.

SOFTWARE, HARDWARE, AND BOOK VENDORS: Micro Cornucopia is establishing a group of reviewers. We would very much like to review your Big Board compatible products for Micro C. Please send material to Review Editor, Micro Cornucopia.

WRITER'S GUIDELINES: All items should be typed, double-spaced on white paper or better yet, on disk. (Your disk will be returned promptly.) Payment is in contributor's copies.

LETTERS TO THE EDITOR: Please sound off.

CP/M is a trademark of Digital Research, Inc.

Copyright 1981 by Micro Cornucopia.
All rights reserved.

MICRO CORNUCOPIA

Sept. 1981

The Journal of the Big Board Users

No.2



*There once was a
Big Board so brisk.
It could eat all the
bits off a disk.*

*It chewed up the bits,
then spit out the pits,
which made feeding it
software a risk.*

Here We Go Again!

Exclusive!

What happens when a Xerox copies a Big Board? Why you get a "Worm", of course! That's right! The Xerox 820 is just a Big Board in disguise.

My informed sources say that last fall Xerox bought non-exclusive rights to manufacture a system based on the Big Board. Xerox re-laid out the board (4 layers) so that it would fit in the cabinet, they dedicated the SIO port B as a printer port, and they set up the disk interface (1771) to handle either 5 or 8 inch. Otherwise, it appears to be all Big Board, right down to the 2.5 MHz clock. The system PIO does the same things on both systems, bit for bit, according to Xerox's documentation.

Xerox had 50,000 orders in hand the day they shipped the first 820, and they expect to recoup all their startup costs by the end of this calendar year. What a market for software and hardware developed around the Big Board. I'll say more about the 820 as information comes in. (I'd give my eye teeth to see a schematic and service manual for the 820.)

Picnic

We had a Saturday noon picnic to celebrate our first issue. It turned out that the Saturday we picked conflicted with every party/birthday/outing/etc. for three states around. But Sandy and I and those who came had six hours of very interesting and mellow conversation.

The knowledge, resources, and excitement among the local group members are terrific. I only wish all of you could have joined us.

The First Issue

Despite the speed of the U.S. Snail, a heartening number of readers have actually received issue no. 1. The responses from these lucky folks have made the daily trip out to our mailbox most enjoyable. The comments have included; 'surprised, happy, delighted'.

Though Micro C is a long way from being a success financially, feedback like this tells us that it is successful in other ways. We like doing it and we really appreciate your response.

Sometimes a dream generates momentum of its own. This one has.

Thanks.

David Thompson
Editor & Publisher

Letters

Supporting A Language

By David Thompson

Dear Sir,

July came and July went by, and my mailbox has completely rusted out due to all that drooling.

Silly me! When I read 'Issue No. 1 will hit the streets during July' I assumed it was July 1981! But now I realize you meant July 1982. I'd better get a stainless steel mailbox or maybe not bother to wait, because the magazine will never get here.

Maybe it went the way of Mitt's Newsletter, the Digital Group Newsletter, and Processor Technology's "Access."

I hope not.

Joe Kish

758 Yucca Ridge Lane
San Marcos, CA 92069

Editor's note:

I called Joe; after all it was the least I could do for his mailbox. And besides, I think it's a great letter! (He did finally receive issue no. 1.)

Sandy and I made a desperate, last ditch effort to get all 500 first issues collated, bound, labeled, sorted and bundled in one afternoon so we could get the first issue in the mail on July 31. We missed the 8 PM deadline at the post office by 15 minutes.

So the magazine was mailed Monday morning, August 3rd. (So much for hitting the streets in July.)

Someday maybe I'll write a book about starting a users group magazine. I could almost write the book about the first issue, and Murphy would certainly be a leading figure. (For those of you who don't know Murphy, he is the one credited with the first voyage of the Titanic.)

Quote from Murphy:

If there is no way
your plan can fail,
you simply don't have
all the information.

Dear Editor,

I bought a bare board version and built it up from scratch. I had to buy about \$80.00 worth of parts beyond what I had around. I have it up and running CP/M and am currently working on packaging it in a terminal-type case with a Ball Brothers CRT. The unit is going to be used for text processing and formatting for a friend's photo typesetter. My other computer is an LSI-11 and I also use

(continued next column)

Throughout these early months of Micro Cornucopia, I have been looking at commercial and public versions of various languages with the hope of finding a semiofficial language for this group.

A common high level language would mean we could pass around source code in something other than assembler. But the language would need to be powerful enough for substantial commercial applications and inexpensive enough that most of the people in the group could afford it.

Letters continued

my H19 with the DEC-20 at work. I think the Big Board is an excellent value and very useful.

I agree that Frank Gentges' idea about the parallel ports is excellent. That would take care of most of the board's limitations. I think your publication has already been worth the price and I suspect that an active users group with a publication will enhance the usefulness of the hardware significantly.

Doug Faunt
PO Box 11142A
Palo Alto CA 94306

Dear David,

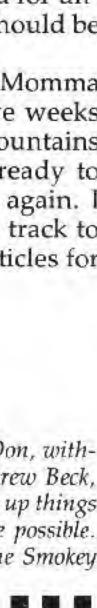
CONGRATULATIONS!!! FANTASTIC!!! You really made it. It looks great and reads great. You are certainly to be congratulated for undertaking such a task that should be helpful to so many.

I hate to mention that Momma and I are just back from five weeks vacation in the Smokey Mountains in Tennessee. I am about ready to get my feet on the ground again. I hope that I can get back on track to help keep the pipe full of articles for future issues.

Don Retzlaff
6435 Northwood
Dallas TX 75225

Editor's note,

What can I say? Thanks again Don, without you and John Jones and Andrew Beck, and the rest of you who are writing up things for future issues this wouldn't be possible. (As for the five whole weeks in the Smokey Mountains, that's just not fair.)



Plus, it would need to produce fast and compact object code, encourage readable source code, and promote structured programming. (Whew!)

I am looking seriously at three languages: Forth, Pascal, and C. Of these three, C is presently leading. One reason is that all the versions I have seen have been upwardly compatible with Bell Lab's C.

Versions of C that I'm aware of:

Small C (Public)
Small C+ (Public)
Tiny C (\$100)
CW/C (\$75)
BDSC (\$145)
Supersoft C (\$200)
Whitesmith's C (\$600)

(The prices are approximate.)

Whitesmith's C is a full blown version of the language. In fact, sources tell me that it was created by three fellows who worked on C for Bell Labs. They left Bell in order to develop and market C for the business and scientific community.

I've heard that BDSC is a competent enough subset to be an option for someone writing commercial applications. It has its own users group and publication. All this for \$145, such a deal. (Lifeboat is offering discounts on quantity purchases of BDSC.)

CW/C is an expanded version of Small C with lots of nice utilities, but I don't know if it is ready to do commercial work. However, it still looks like quite a bargain at \$75.

Tiny C is the only interpreter in the bunch. It also comes in compiler form for about \$300. The only thing I have heard about Tiny C is that it has an excellent manual (and I heard that fourth or fifth hand).

Supersoft's C is new on the market. The ads say that they support 'most' of version 7 Unix. If that includes floating point and pointer arithmetic, then it would be a very credible piece of software, assuming they have taken time to exorcise bugs.

The standard text on C is:

"The C Programming Language"
by Kernighan and Ritchie
Prentice-Hall



Parallel Print Driver Listing

Parallel Print Driver

By John P. Jones

5826 Southwest Ave.
St. Louis, MO 63139

This is a simple parallel printer driver that can be incorporated into any CP/M BIOS.

On first entry, the program initializes PIO port B and the interrupt vector register. The program also modifies the BIOS jump table so that all subsequent calls for list output bypass the initialization routine.

As each character is output to port B, a flag byte is set, indicating that the printer is busy. When the printer is again ready, the PIO does an interrupt. The sole purpose of the interrupt service routine is to reset the 'printer busy' flag. The character output routine tests the flag byte and loops until it is reset. When the flag is reset, a character is sent and the flag is again set.



```
        ORG     CBIOS
                ; STANDARD JUMP TABLE TO
                ; THE SUBROUTINES OF CBIOS
        BVECTR: JP  BOOT
        WBOOT:   JP  CONST
        CONIN:   JP  CONOUT
        DVECTR: JP  OPENPRT
        CONOUT:  JP  CONOUT
        CONIN:   JP  HOME
        SELECT:  JP  SEEK
        SETSEC:  JP  SETSEC
        SETFTR:  JP  READ
        WRITE:   JP  CONST
        TRANS:   JP  TRANS
                ; LIST DEVICE STATUS VECTOR
        OPENPRT EQU   $
                ; PORT B=OUTPUT
                ; F10H CONTROL PORT
                ; INTERRUPT VECTOR B
                ; LOAD VECTOR REGISTER
                ; ENABLE INTERRUPTS
                ; INTERRUPT DEST ADDR
                ; STORE AT VECTOR
                ; CP/M ENTRY
                ; ALL SUBSEQUENT ENTRIES SKIP INIT
                ; NON-ZERO TO A
                ; DECLARE INTERRUPT NOT PENDING
                ; INTERRUPT PENDING?
                ; IF YES, WAIT TIL NOT
                ; ZERO TO A
                ; SET PENDING AGAIN
                ; GET CHAR
                ; PRINTER NEEDS INVERTED DATA
                ; SEND CHAR
                ; INTERRUPT PENDING FLAG
                ; ANY NON-ZERO OK
        INTFND DEFS   1
        PINTFND DEFS   1
                ; PINTFND, A
                ; OUT (0HH), A
                ; RET
                ; JR Z, PRTCHR
                ; XOR A
                ; LD (INTFND), A
                ; LD A,C
                ; CPL
                ; OUT (0HH), A
                ; RETI
                ; PUSH AF
                ; LD A, OFFH
                ; LD (INTFND), A
                ; POP AF
                ; RETI
```

ADS

If you want millions to know what you're doing,
buy a page in Byte.

However, if you:

- need help designing a commercial product
- can provide help on a consulting basis
- need to find a source of...
- want to sell that new BB peripheral we've all been waiting for

Well then, how about an ad in Micro C?

Space Ads

People laugh when we tell them what our space rates are. They stop laughing when they realize that a 1/3 page ad costs about as much as a sack of groceries.

If you are interested in one of our grocery ads or in something larger or smaller, call or write. We'll send a rate card and complete details. The advertising deadline is October 15 for issue no. 3, and December 15 for issue no. 4.

Write or call the editorial office for information.

Want Ads

For a modest 20 cents per word, you could become famous on a budget. (Please include payment with ad.) Where else could you say

WORLD'S GREATEST
PROGRAMMER
503-645-3253

for only 80 cents?

So write it down just the way you'd like to see it. Don't abbreviate the thing to death. List the price if possible and any expected shipping delay.

end

Notes From Garland, Texas

By David Thompson

Clearing up the screen.

The clear-to-end-of-screen command is CONTROL Q, not CONTROL W as indicated in the documentation.

Bringing up stubborn boards.

A number of people have been contacting Jim and me about problems they are having bringing up boards. One of the most common symptoms is a pattern of two characters on the screen or a screenful of random garbage. Either way, it basically means that the board probably didn't finish loading the PFM monitor in RAM so it could try to clear the screen.

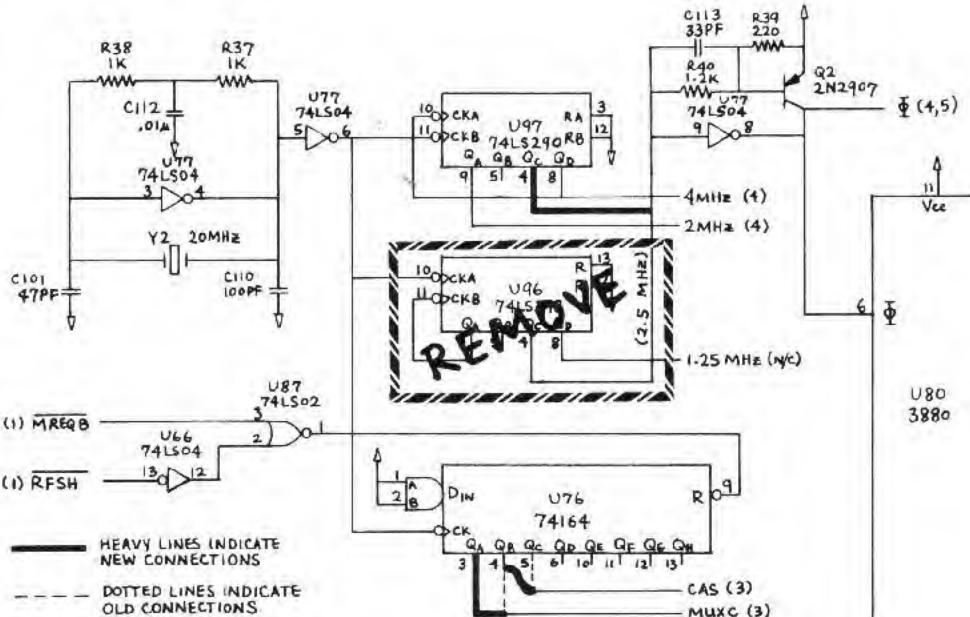
Jim is going to put together information about what they look for when they troubleshoot boards. Hopefully, I will have that in time for the next issue.

Don't forget the 90 day guarantee which completely covers defective parts and boards. Plus, he has been doing out-of-warranty or pilot error repairs very reasonably. Most of the time these charges have been between \$25 and \$50. The maximum so far has been \$75 (the board had to be almost completely resoldered, among other things). That's pretty hard to beat.

Two CP/Ms

I have noticed that some software which runs on one Big Board system will not necessarily run on another. I also noticed that there are two different IDs when CP/M boots.

I called Jim about this and he said that those folks who used the BIOS he sent out with the boards and who did their own incorporation into CP/M have a version which origins the BIOS at EA00. All the folks who bought CP/M already modified for the Big Board have a BIOS starting at E800. The difference has led to some problems with software which depends on having BIOS in a certain place.



Jim said the ready-to-run version has BIOS shifted down 200H because they thought they needed room to store 256 bytes (a double-density sector) in high memory. Then the data could be moved into low memory in 128 byte chunks and accessed. Jim isn't sure whether there is going to be a use for this space but he is concerned that we maintain consistency.

According to Jim, it's easy to make the EA00 BIOS into an E800 BIOS.

Original—.RES.(MSIZE-20)*1024
New—.RES.((MSIZE-20)*1024)-200

Now reassemble the mess and you too can ORG at E800.

By the way, a pretty reliable way to tell which version you have is to look at the ID that's displayed when you boot CP/M. If it just says "60k CP/M version 2.2" then you probably ORG at EA00. If the prompt includes the words "BIG BOARD" then you already ORG at E800.

The separate BIOS (and monitor etc.) disk Jim is shipping with orders now ORGs at E800. If you would like the latest version rather than reassembling BIOS with the modification above, send Jim a disk and \$3.00 for shipping.

4 MHz (Again).

This is an updated version of the 4 MHz mod printed in issue no. 1. This version reportedly does not require special ram. Jim says he has 300ns 4116 working consistently using this mod. The only difference between this one and the previous one is that the CAS and MUXC lines are each moved left one pin on U76 (shift register) so that they change states 50ns earlier. This change means that the system meets the precharge requirements for the slower RAM.

4 MHz Mod Version 2

1. Cut the trace (bottom of the board) to U76 pin 4.
2. Connect the cut trace (MUXC) to U76 pin 3.
3. Cut the trace (bottom of the board) to U76 pin 5.
4. Connect the cut trace (CAS) to U76 pin 4.
5. Remove U96.
6. Connect U97 pin 4 to U96 pin 4.
7. Don't replace U96.

(continued next page)

Disk Drive Motor Control

By David Thompson

CP/M patch for serial printer port.

This CP/M modification redirects the list device output to serial port B. The default data rate is 300 baud. This patch does not force the Big Board to poll any of the handshake lines on port B. Thus, it has no way of knowing if the printer buffer is full. (May or may not be a problem.) This modification is for those who ORG at E800.

Enter the characters inside the quotation marks. <CR> = carriage return.

The patch:

1. Power up the Big Board (BB).
2. Place a CP/M disk with SYSGEN on it, in drive A.
3. Boot CP/M.
4. Enter "SYSGEN" "<CR>"
Displays: SYSGEN VER. 2.0
Displays: SOURCE DRIVE NAME...
5. Enter "A"
Displays: SOURCE ON A,
THEN TYPE RETURN
6. Enter "<CR>"
Displays: FUNCTION COMPLETE...
7. Hit the BB RESET switch <CR>

NOTE: You now have an image of Boot, CP/M, and Bios in RAM starting at 0900H.

8. Remove the source disk from drive A.
9. Enter "M22C7" "<CR>"
Displays: 22C7 00
10. Enter "79"
11. Enter "C3"
12. Enter "18"
13. Enter "F0"
14. Hit spacebar to return to PFM.
15. Enter "M1F90" "<CR>"
16. Enter "47"
17. Enter "EB"
18. Hit spacebar to return to PFM.
19. Place blank disk in drive A.
20. Enter "G100"
Displays: SYSGEN VER 2.0
21. Enter "<CR>"
Displays: DESTINATION DRIVE...
22. Enter "A"
Displays: DESTINATION ON A ...

23. Enter "<CR>"
Displays: FUNCTION COMPLETE ...
24. Enter "<CR>"

The disk now contains a CP/M system that supports CONTROL P (and PIP LST:=) for listings. As mentioned above, the output is on serial port B and is 300 baud.

Editor's note:

To change the baud rate, create F.COM as follows:

1. Enter "DDT" "<CR>"
2. Enter "A100" "<CR>"
3. Enter "MVI A,XX" "<CR>"
4. Enter "OUT 0C" "<CR>"
5. Enter "JMP 0" "<CR>"
6. Enter "<CR>"
7. Enter "G00" "<CR>"
8. Enter "SAVE 1 F.COM" "<CR>"

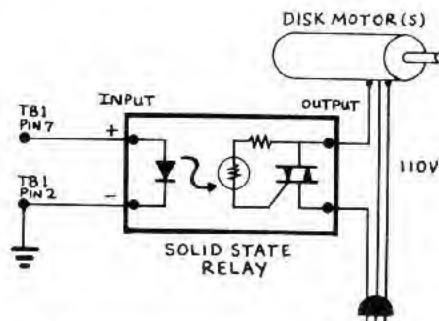
This routine sends a single byte (XX) to the channel B baud rate generator. I am working at 9600 baud so I replace XX with 0E. See the Big Board Theory of Operation for other baud rates.

Once you have completed the baud rate program, simply enter "F" "<CR>" from the CP/M prompt to set the baud rate.

No UPS to a PO Box?

Jim Tanner lists his mailing address as a PO Box but he also has a street address that works for both the post office and United Parcel Service. (The ZIP is different.)

Jim Tanner
Digital Research Computers
2702 Industrial Lane
Suite J2
Garland, Texas 75041
Phone 214-271-3538



Disk AC Control Circuit.

If you're tired of listening to your disk drives grind on hour after hour, here's relief.

The board must have the timer option installed and you must jumper pin 3 to pin 4 and pin 7 to pin 8 on JB2. This supplies the one second interrupt to the Z80. If the Z80 counts all the way to 30 after the most recent disk access then it sends a command to the system PIO to drive the output of U112 pin 2 low.

Terminal 7 on the Big Board power connector is tied to U112 pin 2. This terminal is high (about 4V) when the system is doing a disk access and goes low if there hasn't been an access for 30 seconds.

Simply connect the input of an optically isolated solid state relay between terminal 7 and ground. Then connect the output in series with the AC to the disk drive motors. (But do not connect in series with the drives' DC supply.)

I tried mechanical relays at first, but even the type made to be driven by TTL have problems. Whenever you use mechanical switches to start and stop motors you get interesting transients on the AC line. Interesting transients occasionally cause CPUs to go off picking daisies.

I am now using an ITT solid state relay P6-3DCC-120R5. It has a (P6) package, a 3VDC (3D) input, a 120VAC output with random switching point (120R), and it handles up to (5) amps. It is also small, quiet, and hasn't yet sent the system packing.



Jumpering The Wild Shugart

By David Thompson

Shugart set a new standard for obscurity when they came out with their SA 801 user's manual.

It's not that they don't tell you how to jumper their drives, the only problem is figuring out what they told you. Once you figure it out, don't go back and look at the manual, you'll just get confused again.

So on that note, here's what I figured out.

For drive A, jumper only the following: DC, C, DS1 (Drive Select 1), T2, T3, T4, T5, T6, HL, A, B, T1, 800, Y.

For drive B, change DS1 to DS2. For drive C, change DS1 to DS3, and so on.

For the last 9 months or so, Shugart has been shipping drives with a new circuit board. The new board is completely interchangeable with the old one, but the new one does not use the -5/-15V pin on the DC supply jack (J5). The pin is there but is not connected to anything because the new board does not need -5V.

One way to tell whether you have a new or old style drive is to check the bottom left hand corner on the circuit board. The old drive has a -5V regulator there. On the new one, that corner is pretty empty. Also, the resistance from the -5V pin to ground is infinite on the new boards.

I had one of the new boards but the old documentation so I spent a couple of 'interesting' evenings trying to make sure the -12V I was supplying would be properly turned in-

to -5V on the board. (Oh well, if everyone's documentation were perfect there probably wouldn't be so much need for user groups.)

Note: The following information is from Bill Klevesahl, Shugart's product manager for the SA 800 series.

Test points for both boards.

- 1,2 Amplified read signal
- 5,6,7 Ground
- 10 -Index
- 11 +Head Load
- 12 -Index and Sector Pulses
- 16 +Read Data
- 25 +Write Protect
- 26 -Detect Track 0
- 27 +Step Pulse

Test points on the old board only.

- 3,4 Differential Read Signal (this signal is now hidden inside the new LSI read chip).
- 21,24 -Data Separator Timing (there is no longer a pot to adjust this).

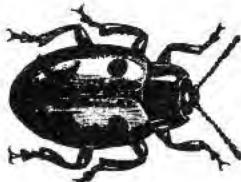
Test points on the new board only.

- 8 +Data Window (for checking FM data separation).

Optional features on the new board.

- Add-trace option TS enables true FM data separation, maintaining synchronization during address marks.
- Add-trace option NFO prevents the head from being forced out past track 0.

BUG



The formatting program listed in issue 1 contains a bug. If the program has a problem accessing a disk in drive B, it reformats the disk in the default drive (A).

Issue 3 will include a revised format program.

Coming Up

Articles you'll be seeing in the future.

- Reverse video cursor
- 5 inch disk interface
- Real time clock routine
- Converting a TV into a real video monitor
- More on the PFM monitor
- Review of 3 assembly language texts
- Bios modifications

Articles we'd love to see.

- Trials and tribulations of bringing up a Big Board
- How you've improved the PFM monitor
- Hard disk interface
- Filling out the second bank with system RAM
- DMA interface
- Double density disk interface
- A graphics display
- A speech generator
- A simple ROM burner
- Interfacing with particular printers etc.
- An in-depth series on CP/M
- Reviews of FIG Forth and Forth 79
- Reviews of BDSC, White-smith's C, CW/C and Super-soft's C
- Computer consulting using a Big Board
- Reviews on peripherals, keyboard, video monitor, power supply, cabinet, disks, etc.
- Other software reviews. Even if you are just borrowing a copy to evaluate, please let us know how you like it.
- Book reviews

If you are immersed in any of these projects, please share your experience with all of us.



Direct Input Routine

By Andrew P. Beck

AB Computer Products
PO Box 571
Jackson, NJ 08527

Assembly Listing

```
F800    E5      SUBR      PUSH HL    :SAVE ADDRESS OF HL%
F801    CD06FO   CALL KBDST   CALL KBDST :GET KBD STATUS
F804    B7      OR A       OR A       :IF A=0 DATA AVAILABLE
F805    CA0EF8   JP Z ISDATA  JP TO DATA SAVE ROUTINE
F808    E1      POP HL    :GET ADDRESS BACK
F809    3C      INC A     :A=FF IS NO DATA, MAKE IT 0
F80A    77      LD (HL),A :STORE 0 IN HL%
F80B    23      INC HL    :DO BOTH BYTES
F80C    77      LD (HL),A
F80D    D9      RET       :RETURN WITH HL% = 0
F80E    CD09FO   ISDATA    CALL KBDIN :GET INPUT CHAR INTO A
F811    E1      POP HL    :GET ADDRESS OF HL% BACK
F812    77      LD (HL),A :STORE DATA, LOW ORDER
F813    23      INC HL    :HIGH ORDER = 0
F814    3600   RET       :RETURN TO BASIC
```

-- Poke the above program into F800+ --

```
500 SUBR = &HF800
510 DATA &HE5,&HCD,&H06,&HF0,&HB7,&HCA,&H0E,&HF8
520 DATA &HE1,&H3C,&H77,&H23,&H77,&HC9,&HCD,&H09,&HF0
530 DATA &HE1,&H77,&H23,&H36,&H00,&HC9
540 FOR I=0 TO 22
550 READ INST
560 POKE SUBR+I,INST
570 NEXT
```

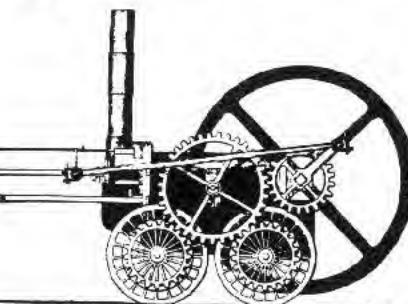
-- Demonstration routine --

```
580 HL% = 0
590 CALL SUBR (HL%)
600 IF HL% = 0 GOTO 590
610 IF HL% = 3 THEN STOP
620 PRINT CHR$(HL%);
630 GOTO 590
```

This routine makes it possible to do direct input with Microsoft basic. First, a machine language subroutine is poked into an unused area of the system monitor.

This subroutine calls the monitor subroutine and the monitor checks to see if an input character is available. If none is available, the HL% is set to zero. If a character is available, it is stored in HL% before a return is executed.

In the demonstration program, a returned character is echoed on the console. If the character is 'C', the demonstration stops.



Something New

DataCast
345 Swett Road
Woodside, CA 94062

I just received issue no. 1 of DataCast and I'm impressed, very impressed. This is a bimonthly magazine for 'major micro systems and telecommunications.' 'Major micro systems' means CP/M in a business or OEM environment and 'telecommunications' means networking.

Jim Warren, guiding force behind the West Coast Computer Faire, is behind this magazine and I suspect it will be around for a long while. Subscriptions are \$18 per year (6 issues).

He is starting with a staff of 19 (if you include the mascot, Sir Lick-A-Lot) and it shows. The first issue is

More Power Supplies

By David Thompson

I just received a catalog from ACDC Electronics and they list a power supply that should power the Big Board and a couple of drives. (Like the Power One, you still have to finagle +12V but that isn't hard, see Issue no. 1.)

Model ETV801 provides:

+5V at 9 amps
-12V at 0.8 amps
+24V at 4.5 amps peak

Price is \$132 (list, single)

They don't mention how they handle over-current protection, but they do indicate that they only have over-voltage protection on the +5V line unless you specify the -1 option. They don't say how much extra you pay for the option.

ACDC Electronics
401 Jones Rd
Oceanside, CA 92054

Power/Mate also has an open frame linear with the same specifications as the ACDC model above, but the PowerMate model ED-132AV lists for \$120 (single).

Power/Mate
514 S River St
Hackensack, NJ 07601



64 pages and about 60 pages of that is copy.

Some first issue articles:

- What is Telidon and Why is AT&T Adopting It?
- Overview of Home Information Services
- A Seminar for Independent CP/M Software Vendors
- Software Documentation Protocols
- An Index to CP/M Software and Vendors

Other Interesting Periodicals

Dr. Dobb's Journal
PO Box E
Menlo Park, CA 94025

Lifelines
1651 Third Ave
New York, NY 10028

Please let us know about your favorite magazines.

Program Storage Above PFM

By Don Retzlaff

6435 Northwood
Dallas, TX 75225

There are numerous times when you want to write a small assembly language program to use as a printer driver or other routine. These small utilities need to reside in high memory so they can operate at the same time as routines which reside in the normal transient program area (starting at 0100H).

Since programs are loaded starting at 0100H, these utilities must load themselves into high memory.

There is a considerable amount of memory available above PFM that is not dedicated to any other use. PFM version 3.3 uses upper memory starting at F000H through F7E6H. The RAM area FF00H through FFC8H is used for data storage. This leaves the memory from F7E7H through FEFFH and FFC9H through FFFFH available for your use. Not all of this space is really available since future releases of PFM could use some of this space.

I recommend that you limit your programs to the following areas: (FA00H through FEFFH and FFE0H through FFFFH).

Moving the program up

In order for your routine to start out as a normal COM file but wind up in upper memory, it has to do a quick shuffle.

1. When the COM file is executed it is loaded into memory starting at 0100H.
 2. Execution starts at 0100H.
 3. The first few statements (starting at 0100H) must copy the routine into upper memory.
 4. An initialization routine may then be executed.
 5. Control is then transferred to the routine or back to PFM.

In order to accomplish all of the above it is necessary to do the following:

1. Write your assembly language routine as follows:
 - a. The origin is set at the desired point where your routine is to reside.
 - b. Your program must start with a short move routine.

- c. An initialize routine usually follows that patches (hooks) your routine into the monitor or PFM.
 - d. Your routine follows.
 - e. The last statement defines the length of the program.

2. Assemble your program.

3. Execute DDT and load your HEX file into memory. Typically this is done as follows:

This will load your program into memory at the desired location (example EA00H). The program will not execute.

DDT will print out starting and ending addresses.

NEXT PC/n
FAXx FA00

4. Using DDT, move the program from upper memory to 0100H.
MFA00,FAx_x,0100
 5. Transfer control back to PFM by typing:
G0
 6. Save the program using the SAVE command.
SAVE 1 NAME.COM
You must save the program in 256

You must save the program in 256 byte blocks. Using '1' will save 256 bytes, '2' would save 512 bytes, etc.

7. The program is now ready for execution as a COM file.

The above procedure may seem long and rather involved but after you have done it a few times you will find it very quick and simple.

— 1 —

PPM Monitor Listing (continued from issue no. 1)

LISTING (continued from issue no. 1)						
F37B	C0	0745	RET	NZ	; ERROR IF > 4 NUMBERS ENTERED	ADD A, 90H
F37C	C5	0746	PARA2:	BC	; SAVE PARAMETER COUNT	DAA
F37D	CD9FF3	0747	PUSH CALL	GETHEX	READ A NUMBER FROM LINE BUFFER	ADC A, 40H
F380	C1	0748	POP BC	RET C	; ERROR IF RESULT OVER 16 BITS	DAA
F381	D8	0749	PARA4:	LD IX, PARAM1	POINT TO PARAM STORAGE AREA	JP OUTPUT
F382	DD217CFF	0750	LD IX, BC	ADD IX, BC	ADD PARAMETER COUNT IN BC	
F386	DD09	0751	ADD	(IX+0), L		
F388	DD7500	0752	LD DD7500	LD (IX+1), H	; STORE DATA RET FROM 'GETHEX'	
F38B	DD7401	0753	LD	LD (IX+1), H		
F38E	FE20	0754	CP	CP		
F390	2BE4	0755	JR	Z, PARA1-\$; GET ANOTHER ITEM IF SPACE	0AH
F392	FE2C	0756	CP	;		0DH
F394	2BE0	0757	JR	Z, PARA1-\$; GET ANOTHER ITEM IF COMMA	0AH
F396	FE0D	0758	CP	CR		0AH
F398	37	0759	SCF			
F399	CD	0760	RET	NZ	; ELSE CHECK FOR CARRIAGE RETURN	EX (SP), HL
F39A	79	0761	PAREN:	A,C	; AND EXIT WITH CY=1 IF NOT	PMSG (SP), HL

```

F39B CB3F A :A=COUNT OF NUMBERS ENTERED
F39D 3C A
F39E C9

0762 SRL A
0763 INC A
0764 RET

0765 ;IGETHEX CONVERTS ASCII TO BINARY AND DOES
0766 ;HIGH LIMIT CHECKS TO LESS THAN 17 BITS.
0767 ;CARRY SET ON ILLEGAL CONVERSION RESULT
0768 ;TERMINATING CHARACTER RETURNS IN A.
0769 ;HL RETURNS WITH 16 BIT BINARY INTEGER
0770 ;HL
0771 ;HL

F39F 210000 0772 GETHEX: LD HL,0
F3A2 180B 0773 LD GNUM3-$
F3A4 0604 0774 JR
F3A6 29 0775 GNUM1: LD B,4
F3A7 DB 0776 GNUM2: ADD HL,HL
F3A8 10FC 0777 RET C
F3AB SF 0778 DJNZ GNUM2-$
F3AA 5F 0779 LD E,A
F3AB 1600 0780 LD D,0
F3AD 19 0781 ADD HL,DE
F3AE DB 0782 RET C
F3AF FD7E00 0783 GNUM3: LD A,(IY+0)
F3B2 FD23 0784 INC IY
F3B4 4F 0785 LD C,A
F3B5 CDBDF3 0786 CALL ASCHEX
F3B8 30EA 0787 JR NC,GNUM1-$
F3B9 B7 0788 LD A,C
F3BC C9 0789 OR A
F3BD 0790 RET
F3BF 0791 ;
F3C0 FE0A 0792 ;ASCHEX: SUB '0'
F3C2 3F 0793 ASCHEX: RET C
F3C3 5F 0794 CP 10
F3C4 D607 0795 CCF
F3C5 0796 RET NC
F3C6 FE0A 0797 RET
F3C8 DB 0798 SUB 7
F3C9 FC10 0799 CP 10
F3CB 3F 0800 RET C
F3CC C9 0801 CP 16
F3CD 0802 CCF
F3CE C9 0803 RET
F3D8 F5 0804 ;
F3D9 1F 0805 ;
F3DA 1F 0806 ;PUT4HS: LD A,H
F3DB 7D 0807 PUT4HS: CALL PUT2HX
F3DC 1F 0808 LD A,L
F3DD C302F4 0809 PUT2HS: CALL PUT2HX
F3DE C302F3 0810 JP SPACE
F3DF 0811 ;CALL PUTNIB
F3E0 F1 0812 ;CALL POP AF
F3E1 E60F 0813 ;CALL PUTNIB; AND AF
F3E2 0814 PUT2HX: PUSH RRA
F3E3 1F 0815 RRA
F3E4 1F 0816 RRA
F3E5 1F 0817 RRA
F3E6 1F 0818 RRA
F3E7 CDE1F3 0819 CALL PUTNIB
F3E8 F1 0820 AF
F3E9 0821 PUTNIB; AND AF
F3EA E60F 0000111B

```

```

F3F1 C9 RET
F3F2 7E FMSG: LD A, (HL)
F3F3 23 INC HL
F3F4 FE04 CP EOT
F3F5 C8 0B45 RET Z
F3F6 CD15F4 CALL OUTPUT
F3F7 18F6 0B46 JR PMSG-$

0B50 ; CRLFS OUTPUTS A RETURN-LTNEFEED-SPACE
0B51 ; TO THE CONSOLE DEVICE
0B52 ; THE CONSOLE DEVICE
0B53 ; CRLFS: CALL PNEXT
0B54 CRLFS: DEF B CR, LF, EOT
0B55 SPACE: LD A,
0B56 LD A, , OUTPUT
0B57 JP

0B60 ; ECHO INPUTS ONE CHARACTER FROM THE CONSOLE
0B61 ; DEVICE, PRINTS IT ON THE CONSOLE OUTPUT AND
0B62 ; THEN RETURNS IT IN REGISTER A WITH BIT 7 RESET
0B63 ; THEN RETURNS IT IN REGISTER A WITH BIT 7 RESET
0B64 ; OUTPUT PRINTS THE CHARACTER IN REGISTER A ON
0B65 ; THE CONSOLE OUTPUT DEVICE AND THEN DOES A CHECK
0B66 ; FOR CONSOLE INPUT TO FREEZE OR ABORT OUTPUT.
0B67 ; FOR CONSOLE INPUT TO FREEZE OR ABORT OUTPUT.

0B69 ECHO: CALL CONIN ; INPUT A CHARACTER AND ECHO IT
0B70 ECHO: PUSH AF
0B71 CALL CONOUT ; CONOUT
0B72 POP AF
0B73 CALL CONOUT ; AF
0B74 CP 'Z'+1
0B75 RET C
0B76 SUB 32 ; CONVERT UFFER CASE TO LOWER
0B77 RET

0B79 ; CONOUT
0B80 ; CONOUT
0B81 OUTPUT: CALL CONST ; SEE IF CONSOLE INPUT PENDING
0B82 CALL Z, OUTP2-$
0B83 JR CONIN
0B84 CALL CR ; SEE IF <CR> WAS TYPED
0B85 CP Z, OUTP1-$
0B86 JR CONIN ; WAIT FOR ANOTHER INPUT CHAR
0B87 CALL OUTP2-$ ; THEN RET TO CALLING ROUTINE
0B88 JR OUTP2-$

0B89 OUTP1: LD (ESCFLG), A ; SET ESC FLAG TO NON-ZERO VALUE
0B90 OUTP2: LD A, (ESCFLG) ; RETURN CURRENT STATUS OF ESC
0B91 OUTP2: OR A ; FLAG TO CALLING ROUTINE
0B92 RET

0B94 ; INCLUDE INTSRV.ASM
0B95 ;
0B96 ;
0B97 ;

```

PFM Monitor Listing (continued)

(continued next page)

(continued on top of page 10)

PFM Monitor Listing

Micro Cornucopia, Number 2, September 1981

PFM Monitor Listing

(continued)

```

1273 PUSH HL, CRTMEM+1
FSEF E5 DE, CRTMEM+1
F5F0 1.10130 LD BC, 24*128
F5F3 01000C LD (HL), :
F5F6 3620 HL, : FILL CRT MEMORY WITH SPACES
F5FB EDB0 LDIR
F5FA E1 POP
F5FB 3E17 HL, :POINT TO HOME CURSOR POSITION
A, 23
F5FD 3277FF LD A, :MAKE BASE LINE# BE 23 AND
F600 D314 LD (BASE), A ;STORE IN SCROLL REGISTER
F602 C9 OUT
RET

1274 HL, :CLEAR REST OF LINE @ HL
F603 E5 PUSH HL, :SAVE CURSOR POINTER
F604 7D A,L
F605 E67F AND 0111111B ;GET COLUMN# COMPONENT OF
F607 4F CURSOR POINTER INTO C
F608 3E50 LD C,A ;CALCULATE HOW MANY CHARS
F60A 91 LD A,B0 ;REMAIN ON CURRENT LINE
F60B 47 SUB C
F60C CD66F6 LD B,A ;CLEAR REST OF LINE @ HL
CALL CLR
POP HL
RET

1275 HL, :CLEAR REMAINDER OF CURRENT ROW
F610 C9 CLREOL: CALL CLREOL
F611 CD03F6 HL, :CLEAR ROW# TO C
F614 E5 PUSH HL, :COPY BASE SCREEN ROW# TO C
F615 3A77FF LD A,(BASE)
F618 4F LD C,A
F619 7D LD A,L
CLRS1: CALL CLRS1:
F61A 17 LD A,H
F61B 7C LD A, H
F61C 17 LD A, H
F61D E61F LD A, H
F61F B9 LD A, H
CLRS2: CALL CLRS2:
F620 2808 LD A, H
F622 CD03F6 LD A, H
F625 CD60F6 LD A, H
F628 1BEF LD A, H
CLRS1-$ JR CLRS1-$
F62A E1 LD A, H
CLRS2: CALL CLRS2:
F62B C9 POP HL, :RESTOR ORIGINAL CURSOR POINTER
RET

1276 DE, -128 HL, :SUBTRACT 1 FROM ROW# COMPONENT
F62C 11B0FF ADD HL,DE ;OF CURSOR POINTER IN HL
F62F 19 LD A,H
F630 7C LD A,H
F631 FE30 CP CRTBAS
F633 DO LD NC ;CHECK FOR UNDERFLOW OF POINTER
F634 263B LD H,CRTTOP-1 ;WRAP CURSOR AROUND MODULO 3K
F636 C9 RET C, CRTBAS
RET

1277 DE, 128 HL, :ADD 1 TO ROW# COMPONENT
F637 11B000 ADD HL,DE ;OF CURSOR POINTER IN HL
F63A 19 LD A,H
F63B 7C LD A,H
F63C FE3C CP CRTTOP
F63E DB LD H,CRTBAS
F63F 2630 LD NC ;CHECK FOR OVERFLOW OF POINTER
F641 C9 RET C, CRTBAS
RET

1278 DEFW BELL ;CTL-G IS THE BELL
DEFW BAKSPC ;CTL-H IS CURSOR LEFT
DEFW TAB ;CTL-I IS TAB
DEFW LFEED ;CTL-J IS CURSOR DOWN
DEFW UFCSR ;CTL-K IS CURSOR UP
DEFW FORSPC ;CTL-L IS CURSOR RIGHT
DCFS 1202 HL, :CLEAR REST OF LINE @ HL
F59C DCFS 1202 HL, :CLEAR REST OF LINE @ HL
F59E BEF5 HL, :CLEAR REST OF LINE @ HL
F5A0 CCFS 1204 HL, :CLEAR REST OF LINE @ HL
F5A2 42F6 HL, :CLEAR REST OF LINE @ HL
F5A4 2CF6 1206 HL, :CLEAR REST OF LINE @ HL
F5A6 C4F5 1208 HL, :CLEAR REST OF LINE @ HL
LFEED 1332 HL, :CLEAR REST OF LINE @ HL
F642 7D HL, :CLEAR REST OF LINE @ HL
F643 17 HL, :CLEAR REST OF LINE @ HL
F644 7C HL, :CLEAR REST OF LINE @ HL

```

```

F$AB E7F5 RETURN
F$AA 11FB DEFW CLRDEOS
F$AC 03F6 DEFW CLREOL
F$AE ECFS DEFW CLRSCN
F$B0 B6F5 DEFW ESCAPE
F$B2 6CF6 DEFW HOMEUP
F$B4 BAFF DEFW STUFF
>0027 1216 DEFW ;CTL-Q IS <CR>
1210 DEFW ;CTL-O CLEAR TO END-OF-SCREEN
1211 DEFW ;CTL-X IS CLEAR TO END-OF-LINE
1212 DEFW ;CTL-Z IS CLEAR SCREEN
1213 DEFW ;CTL-T IS ESCAPE
1214 DEFW ;CTL-V IS HOME UP
1215 DEFW ;CTL-U IS DISPLAY CONTROL CHARS
1217 CTLSIZEQU ;-CTLTAB
1218 ;SET LEAD-IN SEQUENCE STATE
1219 ;FOR XY CURSOR POSITIONING MODE
1220 ESCAPE: LD A,1
1221 LD (DE),A
1222 RET ;SET LEAD-IN SEQUENCE STATE
1223 ;FOR CONTROL CHAR OUTPUT MODE
1224 ;SET LEAD-IN SEQUENCE STATE
1225 STUFF: LD A,4
1226 LD (DE),A
1227 RET ;SET LEAD-IN SEQUENCE STATE
1228 ;FOR CONTROL CHAR OUTPUT MODE
1229 ;CHECK FOR LEFT MARGIN
1230 BAKSPC LD A,L
1231 AND 0111111B
1232 RET Z
1233 DEC HL
1234 RET ;ABORT IF IN LEFTMOST COLUMN
1235 ;BACK UP CURSOR POINTER
1236 ;DO NOTHING IF ALREADY THERE
1237 FORSPC: LD A,L
1238 AND 0111111B
1239 CP 79
1240 RET NC
1241 INC HL
1242 RET ;ELSE ADVANCE CURSOR POINTER
1243 ;TABS ARE EVERY 8 COLUMNS
1244 TAB: LD DE,B
1245 LD A,L
1246 AND 0111100B
1247 ADD A,E
1248 CP BO
1249 ADD NC
1250 RET
1251 LD A,L
1252 AND 1111100B
1253 LD L,A
1254 ADD HL,DE
1255 RET ;PREVIOUS TAB POSITION
1256 ;EXIT IF NEXT TAB COLUMN WOULD
1257 BE PAST THE RIGHT MARGIN
1258 ;ELSE INCREMENT THE CURSOR
1259 ;POINT FOR REAL
1260 ADD RET
1261 RES
1262 OUT
1263 RET ;MAKE LEADIN=2 NEXT TIME
1264 ;VERIFY ROW# BETWEEN Q AND 23
1265 ;MAKE LEADIN=3 NEXT TIME
1266 RETURN: LD A,L
1267 AND 1000000B
1268 LD L,A
1269 RET ;MOVE CURSOR POINTER BACK
1270 ;TO START OF LINE
1271 ;MERGE IN MSB'S OF CRT MEMORY
1272 CLRSCN: LD HL,CRTMEM
>0027 F645 17 ;EXTRACT ROW# COMPONENT OF HL
F646 E61F AND 0001111B
F648 4F LD C,A
F649 CD37F6 CALL DNCSR
F64C 3A77FF LD A,(BASE)
F64F B9 CP C
F650 C0 RET N2 ;IF NOT AT BOTTOM
F651 E5 PUSH HL
F652 CD60F6 CALL CLRIN
F655 29 ADD HL,HL
F656 7C LD A,H
F657 E61F AND 0001111B
F659 3277FF LD (BASE),A
F65E D314 OUT (SCROLL),A
F65F E1 POP HL
F660 C9 RET
F661 7D ;POINT HL TO 1ST COLUMN OF ROW
F662 E680 CLRIN: LD A,L
F663 6F AND 1000000B
F664 065D LD L,A
F666 3620 CLR: LD B,B0
F667 ;STORE ASCII SPACES AT ADD:
F668 23 INC HL
F669 10FB LDN2
F66B C9 CLR-$ RET
F66C 0E20 ;FAKE-OUT CURSOR ADDR ROUTINE
F66E 1817 SETROW-$ ;TO DO HOMEUP ALMOST FOR FREE
F66F ;SET ROW-$
F670 EB UNCONDITIONALLY RESET LEAD-IN
F671 3600 ;STATE TO ZERO BEFORE GOING ON
F673 EB DE,HL
F674 FE01 1 ;POINT HL TO 1ST COLUMN OF ROW
F675 2008 ;GET COLUMN COMPONENT OF
F676 79 SETXY: LD A,C
F677 FESD CF ;GET SECOND CHAR OF SEQUENCE
F678 CO 1 ;=,
F679 FED ;ABORT SEQUENCE IF NOT '='
F67A 3E02 RET LD A,2
F67B 1378 ;MAKE LEADIN=2 NEXT TIME
F67C 3E02 LD (DE),A
F67D 1379 RET
F67E 12 ;MAKE LEADIN=2 NEXT TIME
F67F C9 1381 RET
F680 FE02 ;VERIFY ROW# BETWEEN Q AND 23
F682 2019 CLRIN: LD A,3
F684 3E03 SETROW-$ ;MERGE IN MSB'S OF CRT MEMORY
F686 12 ;MOVE CURSOR POINTER BACK
F687 3A77FF LD A,(BASE)
F68A B1 ADD A,C
F68B D61F SUB -1
F68D D61B 24 ;POINT HL TO 1ST COLUMN OF ROW
F68F 30FC NC,SETR2-$ ;VERIFY ROW# BETWEEN Q AND 23
F691 C61B ADD A,24
F693 F660 OR CRTMEM,SHR,7 ;MERGE IN MSB'S OF CRT MEMORY
F695 67 LD H,A
F696 2E00 LD L,O
F698 CB3C 1396 SRCL
F69A CB1D 1397 RR
F69C C9 1398 RET
F69D FE03 1400 M3TST: CP 3,M4TST-$
F69F 200C JR

```

(continued on top of page 12)

(continued next page)

PFM Monitor Listing (continued)

```

F71F CDABF7 1524 WRITE: CALL READY
F722 CO 1525 RET NZ
F723 CB77 1526 BIT 6,A
F725 CO 1527 RET NZ
F726 06AB 1528 LD B,WRTCMD
F72B 1806 1529 JR RDWT-&

F6A1 79 1402 SETCOL: LD A,C ;ARRIVE HERE ON FOURTH CHAR
F6A2 D620 1403 SUB ;OF ESC, '=', ROW, COL SEQUENCE
F6A4 D650 1404 SETC2: SUB 80 ;MAKE SURE COL# BETWEEN 0 & 79
F6A5 30FC 1405 JR NC,SETC2-& ;NC, SETC2-&
F6A6 C650 1406 ADD A,B0 ;MERGE IN COL# WITH L
F6A7 B5 1407 OR L,A
F6A8 6F 1408 LD L,A
F6A9 C9 1409 RET

F6AD CD72FF 1411 M4TST: CALL DISPLAY ;DISPLAY THE CONTROL CHAR
F6B0 C9 1412 RET ;PASSED IN C
F6B1 1413 F
F6B2 1414 F
F6B3 1415 F
F6B4 1416 F
F6B5 1417 INCLUDE DISKIO.ASM
F6B6 1418 F*****
F6B7 1419 F*
F6B8 1420 F*
F6B9 1421 F*
F6BA 1422 F*
F6BB 1423 F*
F6BC 1424 F*
F6BD 1425 F*
F6BE 1426 F*
F6BF 1427 F*
F6C0 1428 F*
F6C1 1429 F*
F6C2 1430 STSREG EQU WD1771+0 ;STATUS REGISTER
F6C3 1431 CMDREG EQU WD1771+0 ;COMMAND REGISTER
F6C4 1432 TRKREG EQU WD1771+1 ;TRACK REGISTER
F6C5 1433 SECREG EQU WD1771+2 ;SECTOR REGISTER
F6C6 1434 DATREG EQU WD1771+3 ;DATA REGISTER
F6C7 1435 F*
F6C8 1436 RDCMD EQU 10001000B ;READ COMMAND
F6C9 1437 WRTCMD EQU 10101000B ;WRITE COMMAND
F6CA 1438 SKCMD EQU 00011100B ;SEEK COMMAND
F6CB 1439 FINCMD EQU 11010000B ;FORCE INTR. COMMAND
F6CC 1440 RSTCMD EQU 00001100B ;RESTORE COMMAND
F6CD 1441 HLOAD EQU 00000100B ;RDY/WRT HEAD LOAD ENABLE
F6CE 1442 F*
F6CF 1443 RET EQU 009H ;SUBROUTINE RETURN INSTR. OPCODE
F6D0 1444 NMIVEC EQU 0066H ;THE NON-MASKABLE INTERRUPT 15
F6D1 1445 AND B,A ;USED FOR DATA SYNC BETWEEN
F6D2 1446 OR C,D ;THE Z-B0 AND 1771
F6D3 1447 F*
F6D4 1448 F*
F6D5 1449 F*
F6D6 1450 SELECT: LD A,C ;GET UNIT# PASSED IN C AND
F6D7 CP 4 ;CHECK FOR MAXIMUM VALID#
F6D8 D0 1451 RET NC ;ERROR IF NUMBER > 3
F6D9 CDBBF7 1452 CALL TURNON ;MAKE SURE DISKS ARE TURNED ON
F6DA DB1C 1453 A,(BITDAT) IN
F6DB 47 1454 LD B,A ;SAVE CURRENT DRIVE SELECT DATA
F6DC E6FB 1455 AND 11111000B ;MERGE IN NEW DRIVE UNIT# IN C
F6DD B1 1456 OR C ;IN PLACE OF THE CURRENT ONE
F6DE D31C 1457 OUT (BITDAT),A ;TO SELECT THE NEW DISK DRIVE
F6DF CDBCO 1458 CALL FORCE ;TEST NEW DRIVE'S READY STATUS
F6E0 CDBFF7 1459 F*

F72A CDABF7 1531 READ: CALL READY
F72D CO 1532 RET NZ
F72E 06BB 1533 LD B,RDMD ;STORE DISK I/O DATA POINTER
F730 2271FF 1534 RDWR7: LD HL,(FR),HL ;STORE SECTOR# FOR READ/WRITE
F731 216EFF 1535 LD HL,SECTOR ;STORE SECTOR# FOR READ/WRITE
F732 71 1536 INC HL ;SAVE READ/WRITE COMMAND BYTE
F733 23 1537 LD HL,(B) ;SET DISK RE-TRY COUNT
F734 23 1538 LD HL,(C) ;NO INTERRUPTS DURING DISK I/O
F735 F3 1539 LD HL,(D) ;SAVE BYTE AT NMI VECTOR LOCAT
F73D 216600 1540 LD HL,NMIVEC ;IN D FOR DURATION OF READ/WRITE
F740 56 1541 LD HL,(HL) ;LOOP AND REPLACE IT WITH A RET
F741 36C9 1542 LD HL,RECLEN ;B=(HL)
F743 216EFF 1543 LD HL,(B) ;C=1771 DATA REGISTER PORT#
F744 46 1544 LD HL,(LOPTR) ;HL=DISK R/W DATA POINTER
F745 0E13 1545 LD HL,(A) ;GET SECTOR NUMBER
F746 3A6EFF 1546 OUT (SECREG),A ;OUTPUT SECTOR# TO 1771
F747 2E13 1547 LD HL,(C) ;ISSUE FORCE INTERRUPT COMMAND
F748 2E13 1548 LD HL,(D) ;TO TEST HEAD LOAD STATUS
F749 3A6EFF 1549 LD HL,(A) ;GET READ OR WRITE COMMAND BYTE
F750 3A6FFF 1550 LD HL,(B) ;JUMP IF HEAD IS ALREADY LOADED
F751 CB6F 1551 LD HL,(C) ;ELSE MERGE IN HLD BIT
F752 CB6F 1552 LD HL,(D) ;START 1771 DOING IT'S THING
F753 CB6F 1553 LD HL,(A) ;TEST IF COMMAND IS A R OR W
F754 CB6F 1554 LD HL,(B) ;AND JUMP TO THE CORRECT LOOP
F755 CB6F 1555 LD HL,(C) ;HALT
F756 CB6F 1556 LD HL,(D) ;NMI
F757 CB6F 1557 LD HL,(A) ;WLLOOP
F758 CB6F 1558 LD HL,(B) ;HALT
F759 CB6F 1559 LD HL,(C) ;JN
F760 CB6F 1560 LD HL,(D) ;INI
F761 CB6F 1561 LD HL,(A) ;WLLOOP
F762 CB6F 1562 LD HL,(B) ;CALL
F763 CB6F 1563 LD HL,(C) ;BUSY
F764 76 1564 LD HL,(D) ;AND
F765 EDA2 1565 LD HL,(A) ;WLLOOP
F766 ED4F 1566 LD HL,(B) ;CALL
F767 C264F7 1567 LD HL,(C) ;BUSY
F768 C264F7 1568 LD HL,(D) ;AND
F769 C264F7 1569 LD HL,(A) ;WLLOOP
F770 C264F7 1570 LD HL,(B) ;CALL
F771 76 1571 LD HL,(C) ;10111100B
F772 76 1572 LD HL,(D) ;MASK OFF TO READY, NOT FOUND, CRC
F773 76 1573 LD HL,(A) ;WLLOOP
F774 C271F7 1574 LD HL,(B) ;CALL
F775 C271F7 1575 LD HL,(C) ;10111100B
F776 C271F7 1576 LD HL,(D) ;DEC
F777 72 1577 LD HL,(A) ;WLLOOP
F778 FB 1578 LD HL,(B) ;RET
F779 FB 1579 LD HL,(C) ;Z
F780 FB 1580 LD HL,(D) ;HL,RETRY
F781 CB 1581 LD HL,(A) ;MASK OFF AS ABOVE + WRT FAULT
F782 2170FF 1582 LD HL,(B) ;DEC
F783 35 1583 LD HL,(C) ;DECREMENT RE-TRY COUNT AND
F784 2002 1584 LD HL,(D) ;EXECUTE COMMAND AGAIN IF NOT=0
F785 35 1585 LD HL,(A) ;ELSE RETURN 1771 ERROR STATUS
F786 2002 1586 LD HL,(B) ;RETURN IF NO DISK I/O ERRORS
F787 2002 1587 LD HL,(C) ;DECREMENT RE-TRY COUNT AND
F788 B7 1588 LD HL,(D) ;EXECUTE COMMAND AGAIN IF NOT=0
F789 C9 1589 LD HL,(A) ;ELSE RETURN 1771 ERROR STATUS
F790 2160FF 1590 LD HL,(B) ;GET TRACK# FOR THIS OPERATION
F791 18A9 1591 LD HL,(C) ;TRY TO RE-CALIBRATE THE HEAD
F792 CDBBF6 1592 LD HL,(D) ;BEFORE READ OR WRITE AGAIN
F793 18A9 1593 LD HL,(A) ;CLEAR THE DISK CONTROLLER
F794 18A9 1594 LD HL,(B) ;EXIT IF DRIVE NOT READY
F795 18A9 1595 LD HL,(C) ;STORE DISK I/O DATA POINTER
F796 18A9 1596 LD HL,(D) ;STORE SECTOR# FOR READ/WRITE
F797 18A9 1597 LD HL,(A) ;CLEAR DISK CONTROLLER
F798 18A9 1598 LD HL,(B) ;EXIT IF DRIVE NOT READY
F799 18A9 1599 LD HL,(C) ;STORE DISK I/O DATA POINTER
F800 18A9 1600 LD HL,(D) ;STORE SECTOR# FOR READ/WRITE

```

```

F6C3 2806          Z,SEL2-$ ; AND CONTINUE IF ITS READY
F6C5 7B           A,B
F6C6 D31C          OUT
F6CB 3E80          LD
F6CA C9           RET

F6CB 2165FF        HL,UNIT
F6CE 7E           LD A,(HL)
F6CF 71           LD (HL),C
F6D0 FEFF          CF
F6D2 2806          JR Z,SEL3-$
F6D4 23           INC
F6D5 85           ADD A,L
F6D6 6F           LD L,A
F6D7 DB11          IN A,(TRKREG)
F6D9 77           LD (HL),A
F6DA 2166FF        HL,TRKTAB
F6DD 7D           LD A,L
F6DE 81           ADD A,C
F6DF 6F           LD L,A
F6E0 7E           LD A,(HL)
F6E1 FEFF          CP
F6E3 2804          JR 2,HOME-$
F6E5 D311          OUT (TRKREG),A
F6E7 AF           XOR A
F6EB C9           RET

F6E9 CDABF7        HOME:
F6EC CO           CALL READY
F6ED AF           RET NZ
F6EE 326dff        XDR
F6F1 Q60C          LD (TRACK),A
F6F3 CD93F7        RESTOR: LD B, RSTCMD
F6F6 EE04          CALL 1494
F6FB E69C          XDR 0001100B
F6FA C9           AND 10011100B
F6FB C9           RET

F6FB CDABF7        SEEK:
F6FE CO           CALL READY
F6FF 79           RET NZ
F700 FE4D          LD A,C
F702 1505          CP 77
F703 326dff        RET NC
F706 D313          OUT (DATREG),A
F708 061C          LD B, SKCMD
F70A CD93F7        CALL STEP
F70D E69B          AND 1001100B
F70F CB           RET Z
F710 CDF1F6        CALL RESTOR
F713 CO           RET NZ
F714 79           LD A,C
F715 D313          OUT (DATREG),A
F717 061C          LD B, SKCMD
F719 CD93F7        CALL STEP
F71C E69B          AND 1001100B
F71E C9           RET

F793 3A6AFF        STEP:
F796 E603          LD A,(SPEED) AND 000000011B
F798 B0           OR B
F799 CDA3F7        CALL CMDOUT
F79C DB10          IN A,(STSREG)
F79E CB47          BIT 0,A
F7A0 20FA          JR NZ, BUSY-$ ; 1771 AND LOOP TILL=0

F7A2 C9           RET

F7A3 D310          OUT (CMDREG),A ; OUTPUT A COMMAND TO THE 1771
F7A5 CDA3F7        PAUSE: (SP), HL
F7A8 E3           EX (SP), HL
F7A9 E3           EX (SP), HL
F7AA C9           RET

F7AB CDBEF7        CMDDOUT: OUT PAUSE
F7AC E3           CALL (SP), HL
F7AD E3           CALL (SP), HL
F7AE 3ED0          CALL (SP), HL
F7B0 CDA3F7        CALL (SP), HL
F7B3 CB10          IN A,(STSREG)
F7B5 CB7F          IN 7,A
F7B7 C9           RET

F7B8 3E1E          READY: CALL TURNON
F7B9 326cff        LD (MOTOR),A ; KEEP THOSE DISKS SPINNING FOLKS
F7BD CDABF7        CALL PAUSE
F7C0 DR1C          IN A,(BITDAT)
F7C2 CB57          BIT 2,A ; ISSUE FORCE INTERRUPT COMMAND
F7C4 CB           RET Z ; AND EXIT IF STILL TURNED ON
F7C5 E6BB          AND 10111011B ; TEST IF MOTORS HAVE STOPPED
F7C7 D31C          OUT (BITDAT),A ; ELSE RE-ENABLE DRIVE SELECTS
F7C9 CS           PUSH BC ; AND ACTIVATE THE MOTOR RELAY
F7CA 0600          LD B,O ; SET READY LOOP MAX TIMEOUT
F7CC CDDCF7        CALL WAIT
F7CF 2B02          JR Z,TURN3-$ ; WAIT 1/93 SECOND & TEST READY
F7D1 10F9          LD TURN2-$ ; EXIT LOOP IF DRIVE READY
F7D3 0609          LD B,9 ; RE-LOAD MOTOR TURN-OFF TIMER
F7D5 CDDCF7        CALL TURN4-$ ; GIVE ABT 1/10 SEC MORE DELAY
F7D8 10FB          LD TURN4-$ ; WAIT
F7DA C1           POP BC

F7DC DB1B          IN A,(CTC3) ; GET CURRENT CTC3 COUNT VALUE
F7DE 4F           WAIT: LD C,A
F7DF DB1B          LD C, A ; SEE IF CTC3 CHANGED BY 1 COUNT
F7E1 B9           CP C ; AND LOOP UNTIL IT CHANGES
F7E2 28FB          JR Z,WAIT2-$ ; THEN TEST DRIVE READY STATUS
F7E4 18CB          JR FORCE-$

F7E5 0000          ROMEND: DEFW 0
F7E6 FF00          TAIL OF FREE MEM LINKED LIST
F7E7 1521          ORG RAM
F7E8 1522          INCLUDE MEMORY.ASM
F7E9 1523          RET

```

(continued on top of page 14)

(continued next page)

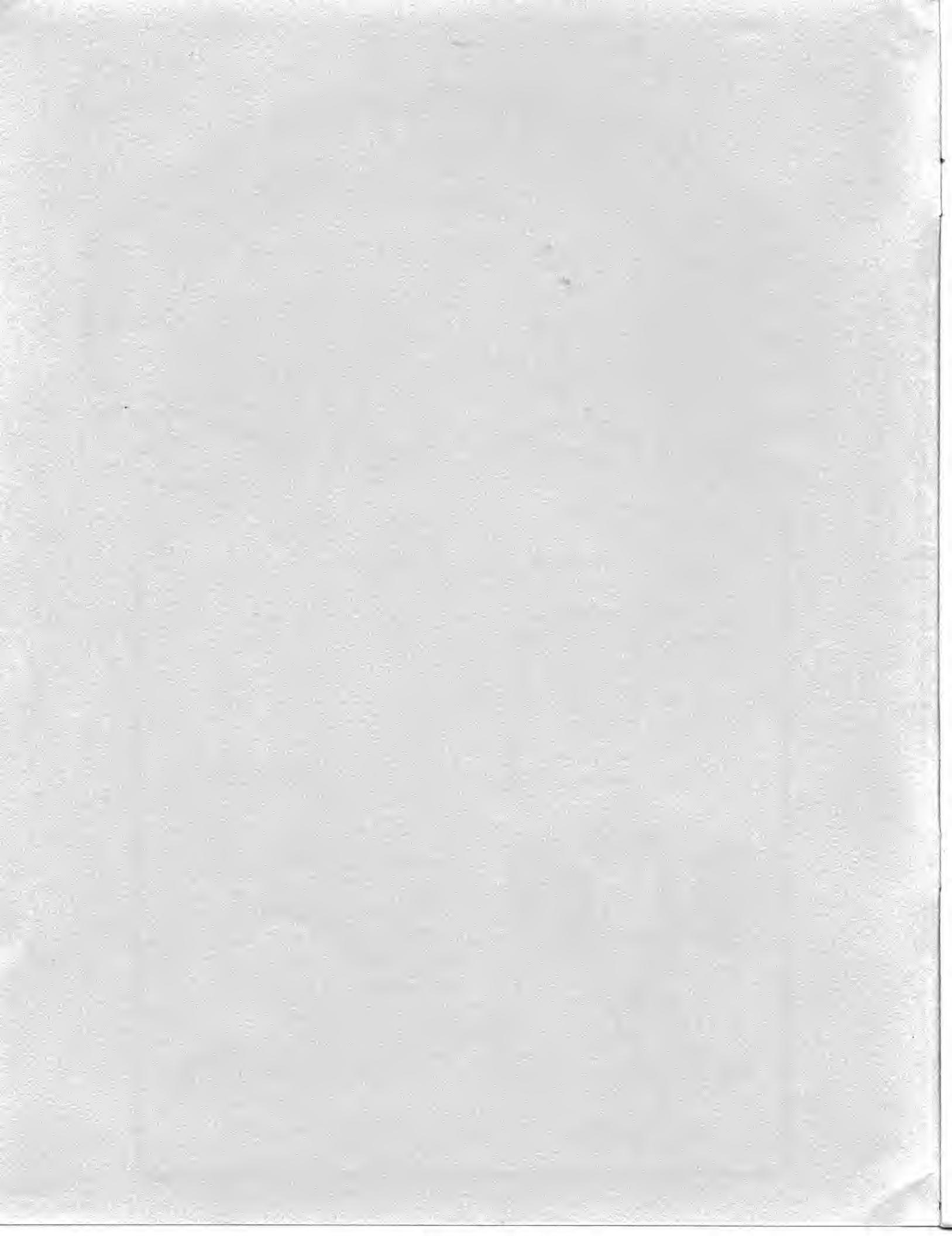
PFM Monitor Listing

(continued)

```

>FF6F ;COMMAND BYTE FOR READS/WRITES
>FF70 ;DISK OPERATION RE-TRY COUNT
>FF71 ;DISK I/O BUFFER POINTER
;
1651 ;*****
1652 ;* STORAGE ALLOCATION FOR 256 BYTE SCRATCH RAM
1653 ;*           DEFNS 1
1654 ;*           DEFNS 1
1655 ;*           DEFNS 1
1656 ;*           DEFNS 1
1657 ;
1658 1659 VECTAB EQU $ ;INTERRUPT VECTOR TABLE STARTS
1660 SIOVEC: DEFNS 16 ;SPACE FOR 6 VECTORS FOR SIO
1661 CTCVEC: DEFNS 8 ;SPACE FOR 4 VECTORS FOR CTC
1662 SYSVEC: DEFNS 4 ;SPACE FOR 2 VECTORS FOR SYSTEM
1663 GENVEC: DEFNS 4 ;SPACE FOR 2 VECTORS FOR
1664 ;GENERAL PIO
1665 ;KEYBOARD DATA INPUT FIFO VARIABLES
1666 ;CONSOLE INPUT FIFO
1667 1668 FIFO: DEFNS 1 ;CONSOLE INPUT FIFO
1669 FIFOCNT: DEFNS 1 ;FIFO DATA COUNTER
1670 FIFIN: DEFNS 1 ;FIFO INPUT POINTER
1671 FIFOOUT: DEFNS 1 ;FIFO OUTPUT POINTER
1672 LOCK: DEFNS 2 ;SHIFT LOCK CHAR+FLAG BYTE
1673 ;
1674 ;STACK POINTER SAVE AND LOCAL STACK FOR INTERRUPT ROUTINES
1675 ;USER STACK POINTER SAVE AREA
1676 SFSAVE: DEFNS 2 ;LOCAL STACK FOR INTERRUPTS
1677 TMPSTK: DEFNS 32
1678 ;
1679 ;
1680 ;'SOFTWARE' VECTORS FOR INTERRUPT SERVICE ROUTINES
1681 ;
1682 1683 TIKVEC: DEFNS 2 ;1 SEC INTERRUPT ROUTINE VECTOR
1684 PINVEC: DEFNS 2 ;PARALLEL CONSOLE INPUT VECTOR
1685 SINVEC: DEFNS 2 ;SERIAL CONSOLE INPUT VECTOR
1686 ;
1687 ;
1688 ;CLOCK-TIMER INTERRUPT VARIABLES
1689 1690 TIKCNT: DEFNS 2 ;BINARY CLOCK TICK COUNTER
1691 DAY: DEFNS 1 ;CALENDAR DAY
1692 MONTH: DEFNS 1 ;MONTH
1693 YEAR: DEFNS 1 ;YEAR
1694 HRS: DEFNS 1 ;CLOCK HOURS REGISTER
1695 MINS: DEFNS 1 ;MINUTES REGISTER
1696 SECS: DEFNS 1 ;SECONDS REGISTER
1697 ;
1698 ;
1699 ;DISK I/O DRIVER VARIABLES
1700 UNIT: DEFNS 1 ;CURRENTLY SELECTED DISK#
1701 TRKTAB: DEFNS 4 ;4 DRIVE HEAD POSITION TABLE
1702 SPEED: DEFNS 1 ;SEEK SPEED FOR 1771 COMMANDS
1703 RECLEN: DEFNS 1 ;SECTOR RECORD LENGTH VARIABLE
1704 MOTOR: DEFNS 1 ;DRIVE MOTOR TURN-OFF TIMER
1705 TRACK: DEFNS 1
1706 SECTOR: DEFNS 1
;
```

end



MICRO CORNUCOPIA

Journal of the Big Board Users Group

P.O. BOX 223
BEND, OREGON 97709